

AD-769 131

NATURAL COMMUNICATION WITH COMPUTERS  
IV

Bolt Beranek and Newman, Incorporated

Prepared for:

Advanced Research Projects Agency

October 1973

DISTRIBUTED BY:

**NTIS**

National Technical Information Service  
U. S. DEPARTMENT OF COMMERCE  
5285 Port Royal Road, Springfield Va. 22151

BBN Report No. 2670

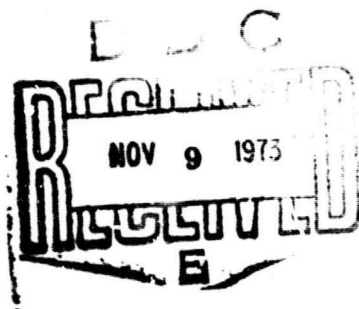
October 1973

AD 769131

NATURAL COMMUNICATION WITH COMPUTERS IV

Quarterly Progress Report No. 12

1 July 1973 to 30 September 1973



The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Advanced Research Projects Agency or the U.S. Government.

This research was supported by the Advanced Research Projects Agency under ARPA Order No. 1967; Contract No. DAHC-71-C-0088.

51-

Distribution of this document is unlimited. It may be released to the Clearinghouse, Department of Commerce for sale to the general public.

Reproduced by  
NATIONAL TECHNICAL  
INFORMATION SERVICE  
U.S. Department of Commerce  
Springfield, V.A. 22161

## DOCUMENT CONTROL DATA - R &amp; D

Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified.

1. ORIGINATING ACTIVITY (Corporate author) <b>Bolt Beranek and Newman Inc. 50 Moulton Street Cambridge, Massachusetts 02138</b>		2a. REPORT SECURITY CLASSIFICATION <b>Unclassified</b>	
		b. GROUP	
3. REPORT TITLE <b>NATURAL COMMUNICATION WITH COMPUTERS IV</b>			
4. DESCRIPTIVE NOTES (Type of report and, inclusive dates) <b>Quarterly Progress Report #12, 1 July 1973 to 30 September 1973</b>			
5. AUTHOR(S) (First name, middle initial, last name)			
6. REPORT DATE <b>October 1973</b>		7a. TOTAL NO. OF PAGES <b>47 52</b>	7b. NO. OF REFS <b>6</b>
8a. CONTRACT OR GRANT NO. <b>DAHC 71-C-0088</b>		9a. ORIGINATOR'S REPORT NUMBER(S) <b>RBN Report No. 2670</b>	
b. PROJECT NO <b>51-</b>		9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)	
c.			
d.			
10. DISTRIBUTION STATEMENT			
11. SUPPLEMENTARY NOTES		12. SPONSORING MILITARY ACTIVITY	
13. ABSTRACT <p>Under this contract, Natural Communications with Computers IV broad based computer science research and development work is performed in areas including; speech understanding systems, speech compression, development of programs and programming aids, techniques for extending computer I/O capabilities, research and development on time sharing systems, and distributed computation. This research program involves the ability to represent knowledge and deal with it in computer oriented terms, requires systems capable of a high degree of man-machine interaction, and draws upon many diverse fields such as linguistics, communications, programming, hardware development, speech recognition, etc. to facilitate fuller use of the enormous potential of natural modes of communication with computers.</p>			

Reproduced by  
**NATIONAL TECHNICAL  
 INFORMATION SERVICE**  
 U.S. Department of Commerce  
 Springfield, VA 22151

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
time sharing systems						
speech understanding						
distributed computation						
programming aids						
BBN-LISP						
Interlisp						
TENEX						
ARPA Network						
Computers and communication						
resource sharing						
speech compression						
vocoders						

## TABLE OF CONTENTS

	<u>Page</u>
I. INTRODUCTION.....	1
II. CONTINUOUS SPEECH UNDERSTANDING.....	5
A. <u>Structure of the November System</u> .....	5
B. <u>Status of the November System</u> .....	8
C. <u>Plans for the Next Three Years</u> .....	13
D. <u>Facilities</u> .....	15
E. <u>Phonological Rules</u> .....	16
III. DISTRIBUTED COMPUTATION AND TEREX IMPROVEMENTS.....	18
A. <u>Distributed Computation - RSEXEC Developments</u> .....	18
B. <u>Peripheral Processor</u> .....	28
C. <u>Monitor Improvements</u> .....	31
D. <u>Subsystem Improvements</u> .....	33
E. <u>TENEX Real Time Interface</u> .....	34
F. <u>Documentation</u> .....	35
IV. LANGUAGES.....	36
A. <u>LISP</u> .....	36
B. <u>Automatic Programming</u> .....	37
V. SPEECH COMPRESSION.....	40
A. <u>Smoothing of Analysis Parameters</u> .....	40
B. <u>Preemphasis</u> .....	41
C. <u>Pitch-Synchronous Analysis</u> .....	43
D. <u>Time-Synchronous Synthesis</u> .....	44
E. <u>Excitation Model for the Synthesizer</u> .....	45
F. <u>Quantization Study</u> .....	46
REFERENCES.....	47

## I. INTRODUCTION

Our Natural Communications with Computers IV ARPA research continued to make progress in the areas of:

1. Speech Understanding
2. Distributed Computation and TENEX Improvements
3. Language Research
4. Speech Compression

Details of the work are included in this our twelfth quarterly technical progress report.

During the past month, work has been devoted to all aspects of the BBN speech understanding system preparatory to the November site visit by the SUR Steering Committee Executive committee. During this period, we have completed initial construction of the major components of the November system (acoustics, lexical retrieval, syntax, semantics, and control) and began shaking out the interactions among them. We have also begun the implementation of a phonological rule component to stand between the acoustic feature extraction component and the higher level components and deal with the systematic variability in pronunciation which can be dealt with by rule. We expect to have the initial version of this component in operation by the time of the November site visit.

All of the components of the system are currently incomplete in many respects and require considerable further development. However, as part of the preparation for the November evaluation,

BBN was required to prepare a document for a Steering Committee meeting to be held in October that would give as accurate a picture as possible of the structure of the November system, its status, and the goals of the BBN effort for the projected five year system. With the exception of two sections dealing with facilities and phonological rules, the substance of this quarterly progress report will be taken from that document. It represents a snapshot of the state of the project as we enter the experimental phase of the project in which we have initial capabilities in each of the individual components and we begin to evolve the techniques for their harmonious interaction.

In our next progress report, we will return to the format of detailed descriptions of the advances made in the individual components of the system.

This quarter our distributed computation project released its second version of the RSEXEC system which enables the RSEXEC user to enter into a distributed file system environment. We also published two RFC's; one describing a proposed file access network protocol and another detailing the TENEX load figures and some observations about the data for network TENEX hosts for July, 1973.

The TENEX efforts continued to be focused on our peripheral processor project. The PDP-11 tool building software made substantial progress with the importation and improvement of a PDP-11 assembler, refinements of our BCPL-PDP-11 cross compiler,

and a BCPL interactive debugger. The computer center has taken delivery of a line printer for our PDP-11 which will be the first peripheral we will remote over the network to our peripheral processor.

The TENEX development effort also included improvements to the TENEX scheduler for increasing processor efficiency under heavy loads. The TENEX EXEC and several TENEX subsystems including FTP and LINK10 were also improved. We also compiled and released a new version of the TENEX JSYS manual.

Our research in computer languages continued its efforts on LISP and also initiated an effort in automatic programming. The LISP work made substantial progress in multiple environments and compiled code overlays with an initial version of the multiple environment LISP available for experimental use. A separate, test system which runs overlaid (swapped) code has been developed. Our automatic programming work has been exploring techniques for program verification, optimization, and synthesis.

In our speech compression research we have investigated several aspects of vocoders including smoothing of analysis parameters, preemphasis, pitch-synchronous analysis, time-synchronous synthesis and excitation models for the synthesizer. The computer simulation program has been expanded to facilitate the testing of many different configurations of speech compression systems. Our study in the past quarter has been explanatory in nature with a view to improve further our

understanding of the roles of the different elements of the vocoder, especially how they relate to the quality of the synthesized speech.

## II. CONTINUOUS SPEECH UNDERSTANDING

### A. Structure of the November System

In the acoustic end of our system, the speech signal is sampled at 20 kHz and stored on a disc file. All subsequent analysis is performed on the digitized signal. Using our recently developed method of "selective linear predictive spectral matching" we perform an LP analysis on the 0-5 kHz region of the spectrum. Presently, almost all our parameters are based on that portion of the spectrum. (The only exception is a parameter giving the spectral energy between 5-10 kHz, which is used for the detection of frication.) The parameters used in our segmentation and feature extraction are based on: energy of the signal, energy of the differenced signal, low-frequency energy, the first autocorrelation coefficient, the normalized LP error, energy-sensitive and energy-insensitive spectral derivatives, fundamental frequency, frequencies of the two-pole model, and poles of the 14-pole model. We have developed an initial set of algorithms for the nondeterministic segmentation of the utterance into a feature or segment lattice. Associated with each segment boundary are confidence measures that reflect the likelihoods of that point in the utterance being a segment boundary and of being a word boundary. Another set of algorithms performs a feature analysis on each of the segments. We have concentrated thus far on the recognition of manner of articulation, e.g. vowel, nasal, lateral, retroflexed, plosive, fricative, voiced/unvoiced. The

only place-of-articulation recognition that we do is performed on the vowels. Confidence estimates for each of the features and for the entire segment are also given.

The resulting segment lattice is processed by a phonological rule component which augments the lattice with segments and possible underlying sequences of phonemes which could have resulted in the observed acoustic sequences. We associate with each added branch a predicate function which is later used by the word matcher to check for the applicability of the given phonological rule based on the specific word spelling and the necessary context. In this manner, the phonological rules are both analytic and generative.

The current lexical retrieval and word matching component makes use of a phonetic similarity matrix for evaluating non-exact phoneme matches, phonologically motivated deletion likelihoods for each of the phonemes in a word, and rudimentary duration cues based on stress marks in the phonemic spelling of the word. Words with three or more phonemes which score above a threshold of match quality are placed in a word lattice and given individually to the semantic component which constructs one-word theories for them, monitors for words that could be semantically related to the given words, and generates events for each detected coincidence between two or more semantically related words. At this point, each word is also checked for matching inflectional endings, and verbs are checked for possible

auxiliaries to their left and at the beginning of the utterance.

These semantic coincidence events are sorted by the control component in order of their likelihood scores and at appropriate times are returned to semantics for the construction of larger semantic theories. In this way, multiple word theories are constructed which consist of semantically related content words which match well acoustically. When such a theory becomes maximal (i.e., semantics has no further words to add to it), it is then passed to syntax for syntactic evaluation. In addition to evaluation syntax picks up further words from the word lattice and proposes words (especially function words) to the word match component to fill the gaps between the words originally provided in the theory. Syntax also monitors for syntactic categories of words which it could use to fill gaps. When syntax completes a constituent (such as a noun phrase) it calls semantics directly to verify the consistency between the syntactic structure of the constituent and the semantic hypotheses for its words.

The control strategy maintains list of active theories, pending events, and proposed words and classes -- all ordered by estimates of likelihood -- and determines which theory/event/proposal to work on next at each point.

Some pragmatic inferences have been identified, but no systematic component has been incorporated. The construction of semantic procedures for answering questions using the data base has not yet been implemented since we have previously done this

once with the LUNAR system and have been devoting our effort instead to the new aspects of the system.

### B. Status of the November System

As of November, all of the principal components of the BBN stage-one speech understanding system will be operating. Working components include: signal processing, acoustical feature analysis, segmentation and labeling, phonological rule application, lexical retrieval and word matching, semantic evaluation and prediction, syntactic evaluation and prediction, and an overall control strategy. The components have been exercised individually and in combination and applied to real data from a number of utterances.

The acoustical analyzer and feature extraction component has been operational for several months now and has been steadily improving in the reliability of its segment identification. However, its current output is rudimentary compared to the output that we anticipate obtaining in the future, and in particular it is generally not as precise or as reliable as the manually produced segment descriptions done by John Makhoul, Dennis Klatt, and Ken Stevens. Its descriptions are generally more vague than the human spectrogram readers and it usually has a higher error rate.

From the opposite direction, the higher level components of the system have been continually improved during the past several

months in their ability to deal with the vague and error-prone output of the acoustic analysis and are now at the level where we can decipher acoustic segment lattices subject to certain limitations on vagueness and amount of error. We have successfully cycled the system on both an ideal segmentation of an utterance (i.e., all and only the correct phonemes) and one of John Makhoul's manually produced segment lattices which contains significant ambiguity and errors in the acoustic analysis. However, we have not yet successfully analyzed one of the more ambiguous mechanically produced segment lattices. Attempts have failed due to fatal phoneme labeling errors which completely block critical word matches (beyond the ability of our error correction strategies to compensate) and to excessive vagueness in the labeling which permits enough extraneous theories that we exceed our current space allocations prior to finding the correct theory.

We have been making rapid improvements during the past few months in both the accuracy and precision of the acoustic segmentation and labeling and in the ability of the other components of the system to deal with ambiguities and multiple theories efficiently. We expect to be routinely deciphering completely mechanically produced segment lattices by the end of the current contract year.

Until a number of additional components have been implemented and current components have been expanded in their

capabilities, we do not believe that a useful performance figure can be achieved for a vocabulary of significant size and a grammar as extensive as ours. The necessary additions include a pragmatics component, speaker normalization operations, an expanded word matching component, more complete phonological rules, and the use of some suprasegmental cues. We believe that acceptable performance rates require all of these sources of information and although we will be able to decipher some sentences with a more limited facility, we do not expect the success rate of such a limited system to be satisfactory.

In view of the goals of the speech understanding project at BBN, we are very pleased with the progress to date. We are getting a good feeling for the problems introduced by the scale of the vocabulary and syntactic flexibility that we want to handle eventually, and are making good progress in developing algorithms for dealing with them. One of our principal concerns has been that the techniques we develop be applicable to problems of realistic size and complexity, and we have been directing our efforts in such a way as to come to grips with these issues of magnitude and scalability as soon as possible.

Each of the components of the current system is imperfect, and even in the final system, the performance of individual components cannot be expected to be without error. Therefore, a major consideration of our effort has been the design of a system organization in which the information available from one

component compensates for the weaknesses or lack of information from another. We are now at the point where the details of these weaknesses and compensating abilities are emerging and experimentation with various interactions is becoming possible.

In summary, the BBN speech understanding project has produced a working system which has cycled on both perfect acoustic analysis data and imperfect and ambiguous manually analyzed data. We have not achieved successful analysis of a completely mechanically generated acoustic analysis, because the extraneous possibilities permitted by the more ambiguous input grow beyond the bounds of our current space and time allocations. However, we expect improvements that we are now making in the acoustic analysis and phonological components to improve this situation. The fact that the system as a whole works together is evidenced by the success achieved with the less ambiguous manual input. We are now well into the experimental phase of the total system development where the tuning and adjustment of the individual components to work in harmony together is one of the central issues, and we are making good progress in this area.

With respect to the 19 specifications in Figure 1.1 of the report, Speech Understanding Systems; Final Report of a Study Group, 1973, our November system will accept continuous speech from a few cooperative speakers (3-10), in a quiet room as well as a moderately noisy computer room, over a good quality microphone, using slight tuning of the system per speaker (to the

extent of measuring average formant ranges), requiring no adaptation by the user, permitting a vocabulary (not acoustically selected) of 260 words, with a natural English syntax of considerable scope and a task permitting complex and specific questions dealing with a large data base of factual information. The current system has no psychological model of the user (beyond a few heuristic biases in the type of sentences which are expected), provides no graceful interaction, does not currently have a measurable success rate, and operates in something like 1000 times slower than real time.

Our target system for 1976 will accept continuous speech from many cooperative speakers with no significant speech pathologies or dialect differences, in moderately noisy environments, over a good quality microphone, using slight tuning of the system per speaker, but requiring only natural adaptation by the user, permitting a vocabulary in excess of 1,000 words, with a flexible natural English syntax and several different semantic tasks. The system will use a pragmatic model of the user and the context of his dialog, provide reasonable interaction, and will aim for not more than a 10% misunderstanding rate on sentences which are otherwise acceptable to a typed-input version of the system. The system will operate in such a fashion that its performance can be used to demonstrate the achievability of equivalent performance in real time when properly implemented on appropriate hardware.

### C. Plans for the Next Three Years

During the ensuing three years of the five year plan, BBN would like to continue the development of a total system for speech understanding -- with emphasis both on the implementation of sophisticated acoustic-phonetic and phonological analysis and on the design of appropriate strategies for using higher level linguistic knowledge to compensate for the indeterminacies and errors in the acoustical analysis. All of the current components of our system are in an immature state and require a great deal of additional work in order to be effective, and we propose to continue to refine and strengthen them. Moreover, as we mentioned before, there are components of the eventual system which we believe are critical but to which we have not yet been able to devote sufficient effort. The following is a list of specific objectives which we would like to pursue during the coming three years.

1. build an acoustic verification component that will take a word and compare it with the acoustic parameters of the signal. This will supplement our current word match component which makes use of previously generated segment labels from the acoustic analyzer. The expanded acoustic verification component will be able to verify detailed structures of inter-phoneme formant tracks, relative durations, etc.
2. explore the possibilities for different ways of organizing the dictionary for sharing common parts of alternative word spellings and perhaps for sharing common parts of the spellings of different words. We would like to conduct several experiments in trying different strategies for cutting down the processing involved in finding closest matches between portions of an acoustic analysis and words in a large vocabulary.

3. integrate some use of prosodic information into the control strategy. We would like to use the results of the U. of Michigan and UCLAC research in prosodics and whatever additional information we may discover as part of our system.
4. continue work on phonological rules, both analytic and generative. We want to continue to develop rules and to incorporate them into the system.
5. begin work on speaker normalization and speaker modelling.
6. expand and improve the parser and grammar. We want to take more context into account during the parsing, make use of more semantic information during the parse, and extend the scope of the current grammar. Our current grammar has been cut back slightly from that of the LUNAR system, and there are further syntactic constructions which we would like to permit which were not contained in that grammar.
7. finish implementing our large vocabulary (1500 words or more). We want to conduct experiments measuring the impact of vocabulary size on the system.
8. implement a large semantic network corresponding to the larger dictionary. We would like to develop ways of constructing, maintaining, and accessing large semantic networks, which will make the system easily movable to other semantic domains beyond the current moon rocks context.
9. expand the pragmatic component. We would like to implement a fairly extensive pragmatic component which would include a model of the discourse for use in dealing with anaphora and ellipsis and a user model for making evaluations of the likelihood of the user making particular utterances in given contexts.
10. study further the mechanisms for evaluating the relative merits of competing theories about an utterance, including the use of evaluation vectors and conditional probabilities.
11. experiment with various control strategies and worry about frameworks in which to compare different strategies. We would like to develop some formal parameters with which to evaluate and compare different possible strategies within this framework.
12. continue our search for algorithms and techniques for coping with the combinatorics inherent in this task.
13. develop a system that we can push a lot of data through. We will need to process large numbers of sentences with different strategies and techniques in order to make

progress during the coming portion of the project. Currently each sentence requires a considerable amount of processing time, and the machine resources which will be required during this portion of the project will greatly exceed what we currently have available. The speech project is currently consuming 25% of BBN's system-A TENEX, and we expect our processing requirements to increase by more than a factor of four. Thus we anticipate needing eventually more computing power than a single TENEX can provide. Solution to this problem may require the acquisition and programming of special purpose hardware such as a fast signal processor.

14. design a projected real-time system. During the final year of the project, when we have gained an improved understanding of the problems and the mechanisms required to solve them, we propose to work out a design for implementing such a system in particular hardware/software combination to achieve real time performance.

#### D. Facilities

##### IMLAC-to-PDP-10 High Speed Interface

During the past quarter we have designed and built a high speed interface for passing data from TENEX to our IMLAC display facility. This 16-bit parallel data path will supplant the 9600 baud serial data path presently used for all data and control, which path will then be used for (ASCII) terminal I/O only. This interface is now being checked out.

The interface has been designed to connect to the PDP-11 which the TENEX project will be using to control a line printer and other peripherals. Until that system is operational, we are using additional logic to connect the "PDP-11 end" to a GP-10 General Purpose Interface presently on the System-B TENEX. When the GP-10-IMLAC system has been completely checked out, the GP-10 will be placed on the main TENEX system. The GP-10 is

essentially a single device interface, which necessitated building a multiplexer for it so as to leave it available for future applications. The GP-10-IMLAC system will be used in the FDP-10-to-IMLAC direction only, which will require minimal software additions to the TENEX system. When the data path is made to operate via the PDP-11, it will be full duplex.

#### E. Phonological Rules

During the last quarter approximately 80 phonological rules were posited. These ran from purely phonological rules of an abstract nature to rules of a highly detailed phonetic kind. Most of the rules appear in two forms: generative, and analytic. Those that do not have straightforward analytic forms make up the generative component. This component will be used to elaborate upon the dictionary entries, which are then fed into the word matching component. The remaining rules, which are in an analytic format, will operate upon the output of the feature extraction component in order to normalize utterance forms before they are fed into the word matcher.

The most common phonological rules which apply to our 1500 word vocabulary have been discovered. Attempts are now underway to incorporate a few of the most useful rules into a running version of the system. Limited attempts are being made to employ a few of the generative rules in an analytic manner with the aid of special techniques.

Continuing research will center upon 6 areas for several months to come. (1) A format convention will be developed so that rules may be conveniently placed in the computer, and then run on the dictionary. The output will then be checked, and the precision and correctness of the rules will then be evaluated. (2) A convention will be developed to express the phonetic detail necessary to the triggering of the rules. (3) Each of the optional rules applies in several environments. For any given rules, the environments are very similar, but differ as to the probability of their application. Representing their different probabilities, either as separate rules or as some set of disjunctions within the rule, is an ongoing matter of investigation. (4) A measure of rule utility will be derived, based not only upon the size of the rule's lexical domain, but also upon the domain's probability of occurrence, and the rule's probability of application once given a suitable occurrence. (5) Rules that apply between words present some difficulties in implementation. This problem is undergoing the first stages of examination. (6) Phonetic data needed to trigger certain rules will be examined, and attempts to extract such crucial information from the signal will be undertaken. Some progress has been made on suprasegmental phenomena, as well as the deletion or addition of segmental forms.

### III. DISTRIBUTED COMPUTATION AND TENEX IMPROVEMENTS

#### A. Distributed Computation - RSEXEC Developments

During this quarter a second version of the RSEXEC system was distributed to the ARPANET TENEX sites. The new version greatly enlarges the capabilities of RSEXEC. An article in the September ARPANET NEWS (NIC #18748) announced the new RSEXEC system to the ARPANET community.

An RSEXEC user may now enter into a distributed file system environment. Within this environment many of the standard TENEX EXEC file system commands have an effect that extends across Host boundaries to include all TENEX Hosts on the ARPANET. To use distributed file system features, a user first defines a "profile" that specifies Hosts and file directories he commonly uses. From that specification a "composite file directory", which includes all files in the directories specified, is created and maintained for the duration of the RSEXEC session. After the composite directory is defined, the user may use standard TENEX EXEC file system commands in a site independent manner, specifying only file name and extension, to access files in his composite directory. In addition, files not in the composite directory may be referenced by using "full pathnames" that include a Host specification; for example, the command

```
+TYPE (File) [SRI-AI]<BJONES>REPORT.OLD
```

prints the file REPORT.OLD in directory BJONES at the SRI-AI

Host.

At the end of a session RSEXEC can be instructed to maintain a permanent copy of the user's profile. By saving his profile a user can use the file system features in future RSEXEC sessions without the necessity of first respecifying his composite directory. If a permanent record of his profile is to be maintained, a copy of a profile file is sent to each Host specified in the profile. By distributing copies in this manner RSEXEC insures that the user can use the file system features of RSEXEC from any TENEX Host without first having to redefine his profile. The profile file itself is transparent to the RSEXEC user, being accessible only indirectly via the profile editing command of RSEXEC.

While RSEXEC attempts to make the network transparent, it helps the user take advantage of the distributed nature of the file system. One way it does this is by allowing the user to increase the "accessibility" of files he considers important by making it easy to maintain multiple copies or "images" of such files at different sites. For example, the following user-RSEXEC dialogue results from an attempt to rename a multi-image file:

```
+RENAME (File) DATA.OLD (to be) DATA.NEW
Rename each Image ? N
Image at [BBN-TENEX]<JONES> ? Y
Image at [I4-TENEX]<RTJ> ? N
Image at [SRI-ARC]<BJONES> ? Y
+
```

Before the RENAME command occurred there were three images of the file DATA.OLD (one each at BBN, I4 and SRI-ARC); after it, there

is one image of DATA.OLD (at I4) and two of DATA.NEW (at BBN and SRI-ARC). As suggested in the previous paragraph user profile files are maintained by RSEXEC as multi-image files.

In distributed systems such as RSEXEC it may often be the case that the peripheral devices needed by a user are not located at the Host where the user's process is executing. To help users cope with this sort of problem the RSEXEC includes a BIND command. With BIND a user can declare that a particular device name is to refer to that device at a Host he specifies. After a device has been "bound" in this manner, subsequent references to the device are directed to the Host specified. For example, a user at the AMES-TIP using an RSEXEC on the BBN-TENEX Host could use the following sequence of commands:

```
+BIND (Device) LPT (to Site) AMES-TIP
++(Port #) 1
+LIST (Files) FILE.ONE FILE.TWO
+COPY (File) FILE.THREE (to file) LPT:
+BIND (Device) LPT (to Site) CASE-10
+LIST (Files) FILE.ONE
```

to produce listings of files FILE.ONE, FILE.TWO and FILE.THREE for himself at device port #1 on the AMES-TIP and a listing of FILE.ONE for a friend at the CASE-10 TENEX. At present the line printer is the only device that can be bound by RSEXEC. We plan to extend the binding capability to include other devices such as DEC tapes.

The new RSEXEC system includes some minor improvements to the network command language service it provides for TIP users. The TIP RSEXEC has been extremely reliable since multi-site

TIPSER, the program that supports use of RSEXEC by TIP users, was installed at BBN-TENEX and USC-ISI (see our last Quarterly Progress Report BBN Report #2607); there is almost never a time when a TIP user is unable to get access to an RSEXEC. We are currently working closely with the TIP group here at BBN to expand the services provided by the TIP RSEXEC system.

Shortly after the new RSEXEC system was distributed the SRI-AI TENEX Host started running the RSEXEC server program on a regular basis. At present eight of the nine ARPANET TENEX sites run the RSEXEC system when they are up and connected to the network. (The exception is the PARC-MAXC Host which intends to run it when it starts running TENEX 1.31, the latest version of TENEX.)

In order to enlarge the scope of the RSEXEC system to include non-TENEX Hosts, the network protocol that supports RSEXEC activity has been redesigned to be less TENEX-specific and, therefore, more easily implementable by non-TENEX Hosts. An unfortunate side effect of this protocol change is that further interaction via RSEXEC with the MIT ITS systems (MIT-DMCG, MIT-AI, MIT-ML) will not be possible until the ITS user and server programs are converted to the new protocol.

The new version of the RSEXEC system solves a deadlock problem involving the exchange of Host status information by the RSEXEC server (RSSER) programs. The problem itself is

interesting in that it did not appear until the system had been running nearly four months. The "lurking bug" in RSSER that caused the deadlock became evident when a relatively minor modification was made to the way the TENE' Network Control Program (NCP) manages network buffer allocations. The appearance of this kind of bug in a program that uses the network demonstrates how sensitive such programs are to even minor NCP changes.

Each RSSER program includes a process (STASER) responsible for exchanging status information with the RSSER programs at other network sites. STASER accomplishes its task by:

1. Periodically waking up and requesting status information from each of the other RSSER programs, and;
2. Maintaining a socket in a "listening" state which other RSSER programs may connect to in order to request status information from it.

In order to prevent the occurrence of so-called "deadly embrace" situations in which two STASER processes at different Hosts each attempt to read status information from one another and thus become "hung" because each is waiting on the other to send the information, STASER is programmed to:

1. Give higher priority to status transmission, and;
2. Abort a transmission to a site if it is simultaneously attempting to acquire status information from the same site. (This prevents two STASERs, each trying to send to the other, from becoming hung as network buffers become full since neither process is emptying them.)

We realized that this approach prevented the deadly embrace situation between any two STASER processes but did not prevent such situations from occurring between "looped daisy chains" of

TIPSER, the program that supports use of RSEXEC by TIP users, was installed at BBN-TENEX and USC-ISI (see our last Quarterly Progress Report BBN Report #2607); there is almost never a time when a TIP user is unable to get access to an RSEXEC. We are currently working closely with the TIP group here at BBN to expand the services provided by the TIP RSEXEC system.

Shortly after the new RSEXEC system was distributed the SRI-AI TENEX Host started running the RSEXEC server program on a regular basis. At present eight of the nine ARPANET TENEX sites run the RSEXEC system when they are up and connected to the network. (The exception is the PARC-MAXC host which intends to run it when it starts running TENEX 1.31, the latest version of TENEX.)

In order to enlarge the scope of the RSEXEC system to include non-TENEX Hosts, the network protocol that supports RSEXEC activity has been redesigned to be less TENEX-specific and, therefore, more easily implementable by non-TENEX Hosts. An unfortunate side effect of this protocol change is that further interaction via RSEXEC with the MIT ITS systems (MIT-DMCG, MIT-AI, MIT-ML) will not be possible until the ITS user and server programs are converted to the new protocol.

The new version of the RSEXEC system solves a deadlock problem involving the exchange of Host status information by the RSEXEC server (RSSER) programs. The problem itself is

three or more processes (i.e., A trying to send to B which is trying to send to C which is trying to send to A, etc.) However, we (unwisely) felt that the occurrence of such chains would be extremely rare and therefore not a real problem. Experience seemed to justify our feeling. For the first four months that the RSSER programs were operational not a single case of a multi-STASER deadly embrace occurred.

In reality, the looped daisy chain situation was (unknown to us) occurring fairly frequently. The deadly embrace did not occur because the NCP buffer management strategy used at the time was to allocate enough buffer space to hold an entire status report when network connections were initially opened; as a result, each STASER in a looped chain was able to safely send its entire report to the "next" STASER in the chain before reading the report requested from the "previous" STASER. (In other words, the NCP's buffer allocation strategy made precaution (b) above unnecessary.) Then, the TENEX NCP buffer management strategy was changed to defer allocation of buffer space until a receiving process first attempted to read from a connection. As soon as this NCP change was installed the looped daisy chain deadly embrace began to occur several times a day; since no STASER on the chain attempted to read until it transmitted its report (sending is considered higher priority) none had allocation to send and a deadlock resulted in which all processes on the chain became hung.

To prevent such deadlocks the new RSSER program uses a simple strategy: two processes are used to maintain status information. One process is dedicated to requesting information; the other, to sending out reports when they are requested.

During this last quarter our work on the RSEXEC system resulted in two RFCs. RFC #535 discussed the possibility of a File Access Protocol. The next major step in the RSEXEC development is to make the TENEX-wide, distributed file system available to executing programs. An important part of that step will be to implement a File Access Protocol. The existing File Transfer Protocol makes it possible for a process on one Host to manipulate files on another Host as "atomic items" (i.e., files may be stored and retrieved in whole, etc.). The intent of a File Access Protocol is to make it possible for a process on one Host to access parts of a file which may be stored on another host. In effect, FTP provides means for a user to have (parts of) file activity of the sort typically initiated at the command language level "slaved" across the network to the site where the file resides. In a similar way, a File Access Protocol would allow file activity of the sort typically initiated by programs at the operating system or monitor level to be "slaved" across the network.

In RFC #546 we report on TENEX load figures for the month of July 1973. In that RFC the average weekday load for the ISI and BBN Hosts is plotted as a function of time of day. From these

plots we made some observations regarding the working habits of the ISI and BBN user communities. The load data is obtained as a side effect of the activity of the RSEXEC system. Figure 1 is the average weekday, load figures for the network TENEX Hosts for September 1973.

LOAD DATA FOR ARPANET TENEX SITES  
 FROM: SEPTEMBER 1, 1973  
 TO: SEPTEMBER 30, 1973

-ND- NO DATA FOR SITE

\* FEWER THAN 25.0% OF POSSIBLE SAMPLES AVAILABLE FOR AVG

WEEKDAY HOURLY LOAD AVERAGES

BBN	CASE	CCA	I4	ISI	NIC	SRI-AI	UTAH
0000							
.62	.94	1.53	.75	1.28	.77	.80*	1.46
.52	.87	1.45	.59	1.12	.58	.79*	1.13
.41	.81	1.27	.38	.96	.67	.70*	.86
.39	.75	1.11	.41	.57	.34	.52*	.72
.46	.74	1.11	.33	.61	.20	.31*	.77
.46	.75	1.09	.17	.61	.24	.09*	.52
0600							
.63	.74	1.38	.36	.61	.70	.40*	.52
.99	.85	1.29	.40	.75	1.17	.07*	.40
2.29	1.01	1.59	.45	1.37	1.44	.07*	.26*
3.65	1.31	2.01	.92	2.31	1.62	.16*	.58*
4.69	1.57	1.79	1.91	3.90	3.49	.65*	1.55
3.72	1.66	2.12	2.47	5.77	3.63	.63*	2.41
1200							
4.24	1.61	2.21	3.13	7.52	4.02	.64*	3.73
6.03	2.07	2.63	3.61	7.56	4.00	1.76*	3.26
6.24	2.28	2.33	2.00	6.75	3.83	.96*	3.63
6.11	1.72	2.23	3.04	5.16	3.63	1.47*	4.74
4.57	1.39	1.86	3.84	5.48	3.19	1.22*	3.91
2.09	1.13	1.62	3.19	4.97	3.39	2.11*	3.63
1800							
1.66	.97	1.33	2.66	3.90	3.47	1.76*	3.92
1.50	1.05	1.20	1.96	2.77	2.81	1.10*	2.11
2.19	1.05	1.52	.82	2.96	2.26	.54*	1.62
1.71	1.11	1.73	.71	2.52	1.82	.67*	1.31
1.35	1.00	1.65	.71	2.13	1.36	.75*	1.17
.96	.95	1.74	.68	1.58	1.27	.81*	1.53

Figure 1

In July we participated in a panel session at the 1973 Summer Computer Simulation Conference titled: "Gaming-Simulation with Computer Communication Networks". The session was an interdisciplinary one in which experts in simulation, gaming, teleconferencing and distributed computing discussed the feasibility of combining their technologies to provide tools to assist decision makers in government and industry in dealing with the urgent problems of energy use, trade balance and monetary stability in the international arena. The session itself was part of an effort which has since developed into a joint U.S.-Japan project known as GLOSAS (for Global Systems Analysis and Simulation). The initial focus of the GLOSAS Project will be the energy crisis facing both the U.S. and Japan. The general objectives of the project are "to promote simulation methodologies on a global scale (via international computer networking), to solve worldwide problems centered on energy issues, and to develop better understanding among policy makers in each country on these problems as well as furthering scientific and cultural exchange". (quote from draft project proposal for GLOSAS entitled "Global Systems Analysis and Simulation of Energy, Resources and Environmental Systems"). We view GLOSAS as an exciting and important project and plan to continue our participation in it.

## B. Peripheral Processor

We are in the process of moving character-oriented device processes from the TENEX monitor into a PDP-11 ARPANET mini-host peripheral processor. This will improve the accessibility and maintainability of these peripherals, while economizing on the 600 microseconds per character processing currently required of the PDP-10.

The first step in this project is providing an adequate level of software production support for the PDP-11. Since TENEX provides the most supportive programming environment available, this will be done by cross-assembly, cross-compilation, cross-debugging, and cross-network bootstrap loading: all software production work will be done inside TENEX.

Accordingly, we surveyed all available PDP-10 to PDP-11 cross-assemblers, and adopted the PAL-11X generated by the Massachusetts Institute of Technology Artificial Intelligence Lab. This was the most powerful cross-assembler we found, as it provides very talented macro facilities and pseudo-ops. However, it only provides generation of absolute code, making it impossible to link together separately-assembled modules. During this quarter, we modified the assembler to generate relocatable code which is linked using the standard Digital Equipment Corporation subsystems LOADER or LINK10. The modified assembler has been returned to MIT, distributed to the ILLIAC Project at NASA AMES Research Center, and will be distributed to all ARPA

TENEX sites as soon as we have updated the PAL manual to reflect our changes.

Of course, we want to program the PDP-11 in a high level language in order to reap the benefits of software which is easier (less expensive) to generate, understand, debug, modify, and export. The language we selected is BCPL, (Basic Compatible Programming Language) which was developed at the Massachusetts Institute of Technology Project MAC. This language currently supports the signal processing functions of our Speech Understanding and Recognition System, and is a particularly good choice because it is quite general, extremely simple, and relatively efficient.

In our last Quarterly Progress Report, (BBN Report #2607), we reported an initial version of the BCPL cross-compiler which was generated by replacing the PDP-10 code output module with a PDP-11 code output module. The output of that cross compiler, however, was disappointingly inefficient in its use of (scarce) PDP-11 core memory.

This quarter we recoded the BCPL translator module to include two stages of local optimization: register and storage allocation, and operations which can be combined or simplified. The new PDP-11 cross-compiler generates very efficient code, and it will be distributed to all ARPA TENEX sites as soon as we complete the BCPL manual. In the meantime, this compiler was delivered to the University of Utah for testing and evaluation in

connection with their PDP-11 display system development. This compiler generates PAL-11X source code which may be separately assembled, then link-loaded with other BCPL or PAL modules.

The new line printer is now connected to the PDP-11, and a number of timing, testing, and demonstration programs have been written in BCPL and run on the PDP-11. Of course, the line printer will not be available over the network as a peripheral to System A and System B until our IMP is delivered next month. (There are currently no IMP-host interfaces available for attaching the PDP-11 to the network.)

Debugging these programs has been quite painful, as no debugging aids were available, and problems were being concurrently uncovered in the compiler, the assembler, the PDP-11 processor, and in the line printer hardware. The need for debugging aids is critical, and we are now exerting effort in that direction. BDDT, the interactive, symbolic debugger for PDP-10 BCPL programs has been operational for several months and is now ready for distribution to all TENEX sites. It is more general and powerful than DDT, as it provides stack untracing features, execution of arbitrary command strings when encountering a breakpoint, and display of the original BCPL source statement corresponding to the program location under scrutiny.

The third-level network protocol has now been defined to permit BDDT to serve as a cross-network debugger which will debug

BCPL code in any machine which provides the protocol interpreter. This protocol is extremely simple, permitting easy implementation of a small debugging module for any machine. Of course, all source and symbol files will remain on the TENEX system where BDDT is running.

### C. Monitor Improvements

Measurements performed on the BBN-TENEX system running with 192K of memory under heavy load revealed that typically 30-35% of available processor cycles were being expended in the scheduler. The problem tended to intensify with smaller amounts of core due to increased scheduling frequency.

More detailed measurements were made to pinpoint those sections of the scheduler which were heavy consumers of processor time. These indicated that redesign of certain sections of the scheduler was called for and, in addition, other system modifications were needed to reduce the scheduling frequency.

A new version of TENEX, with a redesigned scheduler and other system modifications is now in operation at BBN and we are currently evaluating the performance. Measurements show that under typical mid-day load with 176K of core currently available, we now expend 15% scheduling. The measurements have also indicated that further improvements are possible and an updated version has been prepared but has not been evaluated at this writing.

Our program of system measurements has also revealed weaknesses in the core management logic presently used in TENEX and efforts are now underway to improve this and, hence, system performance when running with relatively small amounts of memory.

Another feature added to TENEX during this quarter allows more convenient use of high-speed terminals. This feature counts the number of line-feeds output since the last input operation to prevent terminal output from scrolling out of view before the user has an opportunity to read it. Just before outputting the line-feed that would cause the count to reach the page length terminal parameter, a bell is printed and output is suspended by waiting for any character to be typed in. The character so typed is thrown away and output continues (remember that any input operation zeroes the line-feed counter). To prevent confusing type-ahead with the continuation character, output is not suspended if type-ahead exists (i.e. the input buffer is not empty). This feature is implemented by use of a new terminal type (number 10 octal). The EXEC command "TERMINAL (TYPE IS) SCOPE" allows the user to specify this terminal type. The new "LENGTH (OF PAGE IS)" command allows the page length of a terminal to be specified. Note that use of this command is not restricted to display terminals. Other terminal type names have also been added to combine terminal type "SCOPE" and the page length of particular brands of display terminals.

#### D. Subsystem Improvements

Version 1.51 of the TENEX EXEC has been distributed to sites on the ARPANET. This version contains (in addition to those described in BBN Report Number 2607) commands for redirecting input and output files ("REDIRECT" and "DETACH") and determining ("TRMSTAT") the parameters associated with the terminal as seen by a user program running under the EXEC.

Files listed by the "DIRECTORY" command are now flagged with ;S if the file is a scratch file and ;T if it is a temporary file. If a file has been declared ephemeral, a ;E flag will also be included. The distinction of scratch files and temporary files is an aid when the user is attempting to decide which of the options to "EXPUNGE" to select -- previously both scratch and temporary files were printed with ;T thus creating confusion.

In the past some amount of difficulty was experienced with the accounting system if the time and date were specified incorrectly when the system was restarted. To help prevent these errors, the EXEC checks the time typed in by the operator against the last write date of the FACT (accounting) file. If the specified time is not within eleven hours after the last FACT file entry, the time given by the operator is typed out in a verbose format and he is asked to reconfirm it.

Several improvements were made to the FTP (File Transfer Protocol) subsystem this quarter. First, a site-dependent option

was provided to permit access by an anonymous user, i.e. a person at another site who wishes to access files without going through the FTP "login" sequence. This user is permitted to access only those files which are specified to be accessible to everyone. This option is currently in use at BBN and at the University of Utah.

FTP has also been changed to make use of the account validation mechanism used for system login. A remote user may either make use of his default account number for FTP service charges, or he may supply an account number. FTP then verifies that this user may access the account in question.

LINK10, the new loader provided by Digital Equipment Corporation, is now fully operational on TENEX. The new system building procedure takes advantage of LINK10 to achieve a large speedup in generation of a new system.

#### E. TENEX Real Time Interface

The analog to digital converter and associated equipment have been re-aligned to improve the linearity and reduce the DC offset. The noise floor has been measured and found acceptable for use in the speech research project.

An addition has been installed to allow sampling instants to be specified by external events rather than by the internal clock. This permits an experimenter to (for instance) avoid

distortion due to tape stretch by recording a clock track in addition to his data on a multi-channel tape recorder. The clock signal is then used to trigger sample taking when the data is digitized.

#### F. Documentation

The new JSYS Manual was completed this quarter. It has been considerably expanded to document the user interface to the ARPANET. A number of helpful examples of simple, common programming sequences are included to assist the novice TENEX programmer. In addition documentation is provided for all the system calls which were installed since the publication of the previous JSYS manual. The previous manual is currently available on-line at the NIC, and the current manual will be made available to the NIC to use as an update.

The TENEX TECO Manual has been updated to reflect all recent changes to the program itself. The manual will be available soon.

#### IV. LANGUAGES

##### A. LISP

We have made substantial progress on multiple environments and compiled code overlays.

The initial version of Interlisp with multiple environments is available for experimental use. It includes all the required modifications to the interpreter and the machine language SUBR's, as well as the required changes to the break package, the editor, and PRETTYPRINT. All of the basic stack manipulation functions are implemented. The remaining tasks are the compiler modifications and the stack compactifier.

We also have developed a separate test system which will run overlaid (swapped) code. The following areas of the project have been completed:

1. MKSWAP and MKUNSWAP  
functions to make resident compiled code be swapped or vice-versa.
2. The internal mechanisms to call and return from swapped code; and BRSFIX, the routine to adjust the swapped code buffer and base register when returning to a swapped function.
3. SWAPIN  
the swapper which is responsible for moving code in and out of the buffer and keeping track of what is in the buffer.
4. SETSBSIZE  
a function for declaring how much of the resident address space should be reserved for the swapping

buffer. This allows convenient optimization of the buffer size.

5. BRREST  
a routine to restore the buffer and base register from control tables. This permits clearing the buffer area before saving the state of the system (SYSOUT, MAKESYS) and using the buffer area for special internal purposes such as garbage collection.
6. The changes required to other functions such as FNTYP which is aware of two new types, and the non-standard control transfers RETFROM, RETEVAL, etc. which must restore the base register and buffer for the function returned to.
7. Changes to the block compiler to permit block compiled code to be self-relocating.

There are three areas that need more work prior to an Interlisp release with overlays. They are:

1. MAKESYS and SYSOUT  
these are in progress but incomplete.
2. Garbage collector modifications to reclaim from swapping space. The trace from swapping space is done.
3. Merging the multiple environment system with the overlay system.

## B. Automatic Programming

Among the "lowroad" problems mentioned in a recent automatic programming report by Balzer [1], is that of data representation theory. Specifically, what perturbations can be made to a representation without affecting its real use and how is a data representation interdependent with procedures?

In BBN Automatic Programming Memo 3 [2], we have developed an axiomatic treatment of data structures at basic representational levels, and we demonstrate that it has the following benefits:

1. It permits one to change the underlying representation of a data type while leaving the properties of the data type invariant.
2. It permits one to define the semantics of data definition facilities and of the use of structured values in programming languages. (Consequently, program verification techniques can be applied to programs that operate on structured values).
3. It permits one to exhibit techniques for synthesizing data representations that meet a set of prescribed behaviors specified for a data type.

BBN Automatic Programming Memos 2 and 4 [3,4] explore heuristic techniques for synthesizing inductive assertions needed to prove programs correct. These techniques hopefully bring practical program verification systems closer to reality by improving the range of useful verification tasks for which the machine can assist.

BBN Automatic Programming Memo 1 [5] contributes to the theory of program optimization. To carry out significant program optimization it is frequently necessary to know what properties hold at each program unit. It is shown that it is possible to compute these properties when they form a "well-founded partially ordered set".

Further work in automatic programming is being directed at the problem of program synthesis. This stems from our belief that while short term software productivity gains may come from the practice of programmer disciplines (such as structured programming and improved team management techniques), intermediate and long range productivity gains will come from handing to the machine the task of filling in details in the writing of programs.

## V. SPEECH COMPRESSION

When listening through loudspeakers, the quality of the synthesized speech seems to be quite close to that of the original. However, a careful evaluation using earphones indicates that even the speech synthesized with no quantization and using hand-computed pitch exhibits some amount of "unevenness", "roughness" and "hollowness", when compared with the original speech. Quantization of parameters only degrades the quality further. This is why we devoted considerable effort in the past two quarters to improving the quality of the synthesized speech by several means. While this study has resulted in some improvement and has opened up several new possibilities to explore, further work is clearly warranted before fully satisfactory synthesis can be achieved. Quantization properties of many sets of analysis parameters are currently being investigated.

### A. Smoothing of Analysis Parameters

A study of the time series of analysis parameters (energy, pitch and reflection coefficients) has indicated that occasionally some parameters change rather rapidly, possibly contributing to the "roughness" or "unevenness" that is sometimes present in the synthesized speech. These rapid variations can be smoothed out by a proper low-pass filtering. We used a three-point smoothing filter with weights 0.25, 0.50 and 0.25.

Smoothing was done just prior to interpolation at the synthesizer. Genuine jumps in parameter values were preserved by not smoothing in transitions between voiced and unvoiced sounds, and when parameter changes exceeded preset thresholds (10 Hz for pitch and 3 dB for energy). Identical low-pass filters were used for smoothing pitch, energy and reflection coefficients (or area ratios).

Several synthesis experiments were performed using smoothed parameters and an informal listening test was arranged to compare synthesized speech with and without smoothing. The quality of speech improved in some instances with smoothing insofar as the "unevenness" observed in synthesis without smoothing disappeared. But, smoothing also made the synthesized speech sound less "crispy" or more "smeared" in some instances. At the present time, it is not clear whether the situation can be improved with the use of a more sophisticated low-pass filter for smoothing.

#### B. Preemphasis

In our Quarterly Progress Report #10 (BBN Report No. 2544, April 1973) it was maintained that a proper preprocessing of the speech signal, such as preemphasis, is required to reduce the short-time spectral dynamic range and thereby to improve the efficiency of parameter quantization. In the past quarter, we have investigated, using the quality of the synthesized speech as the criterion, the different types of preemphasis: (a) fixed

preemphasis and (b) adaptive optimal preemphasis of first and second order. The adaptive preemphasis is computed by using the linear prediction for each frame. As mentioned in Quarterly Progress Report #10, BBN Report Number 2544 the adaptive preemphasis data can be transmitted using only 2 bits per frame for the first-order case and 4 bits per frame for the second-order case. Our synthesis experiments indicated that the adaptive optimal first-order preemphasis is preferred to both fixed preemphasis and adaptive optimal second order preemphasis. Fixed preemphasis of 6 dB per octave (simple differencing), for instance, may be satisfactory for most voiced sounds but may turn out to be undesirable for unvoiced sounds.

Our earlier study summarized in our Quarterly Progress Report #10 had led us to believe that the second-order optimal preemphasis is more desirable than the first-order case as the emphasized speech has a spectrum that is relatively flat thereby permitting a good quantization accuracy. However, we have since found that when post-emphasis is done at the synthesizer, distortions are introduced into the spectrum. Specifically, the first formant is affected the most for voiced sounds; its frequency is often lowered, thereby producing a nasal quality or enhanced low-frequency buzz in the synthesized speech. It should be mentioned that such distortions occur even without any quantization. A primary reason for this phenomenon is that the second-order preemphasis often flattens the spectrum by eliminating the prominent formant in the speech. Upon

postemphasis, this formant is not restored exactly. This was highlighted in our experiment where we observed that the cascaded filters of total order 12, containing an optimal second-order preemphasis filter and a suboptimal 10th order linear prediction filter, led to an inferior speech quality than a 10th order optimal linear prediction filter with no preemphasis on the speech signal.

With optimal first-order preemphasis, only one real pole is removed from the input speech spectrum. This does not result in any perceivable distortion in the synthesized speech. Thus, in our future investigations, only the adaptive optimal first-order preemphasis will be used.

### C. Pitch-Synchronous Analysis

Our purpose for investigating pitch synchronous analysis was to see if it would yield any perceivable improvement in speech quality. Hand-extracted pitch data was used to mark the analysis windows of length approximately equal to one pitch period and with the window ends lying on zero-crossings of the speech signal. It was found that if the analysis window does not have its ends on the zero-crossings of the speech, then the estimated spectrum has a shape much distorted from what one would obtain from the Hamming-windowed speech of, say, 20 msec duration. The analysis parameters, energy and reflection coefficients, were extracted and transmitted pitch synchronously. Both

autocorrelation and covariance methods were used in our study. Informal listening tests indicated that there was not sufficient improvement in speech quality to justify the added complexity of pitch-synchronous analysis.

#### D. Time-Synchronous Synthesis

In all our earlier experiments synthesis was done pitch-synchronously. Parameters received at the synthesizer were interpolated to get pitch-synchronous data and were then used to update the synthesizer. In the time-synchronous synthesis, parameters of the synthesizer (except pitch) are reset time-synchronously. In our experiments on time-synchronous synthesis, analysis parameters were extracted and transmitted once every 5 msec. No interpolation was done at the synthesizer. That is, synthesizer parameters were reset also at 5 msec intervals. It was found that the speech quality obtained is clearly better, and closer to that of the original speech. However, transmission rate associated with such a system would be excessive. This study has indicated in particular that our interpolation schemes need to be improved if the speech quality is to approach that of the time-synchronous analysis-synthesis system with a large frame rate and no interpolation.

### E. Excitation Model for the Synthesizer

In one of our experiments, synthesis was done using the unquantized error signal for the excitation of the synthesizer. The synthesized speech had excellent quality, and it was identical to the original speech when both energy and reflection coefficients were not quantized. This result has two implications. First, synthesized speech with good quality can be obtained employing a high bit-rate vocoder that transmits quantized error signal. Secondly, for the low bit-rate systems, other types of excitation models may be explored with a view to improve the speech quality. Presently, we use the excitation model whose output is either a sequence of pulses separated by the pitch period or a white noise sequence. Between this model and the error signal, there may exist a number of different models with varying complexity. We chose one such model, Rosenberg's polynomial excitation [6], to test its appropriateness in our synthesis. The signal obtained from this polynomial function was preemphasized using a second-order optimal preemphasis in order to flatten its spectrum. The spectrally flattened signal was then used as the excitation for the synthesizer. (Spectral flattening of the excitation signal is necessary since the linear prediction filter already includes the effect of the glottal spectral shaping.) It was found that the quality of the synthesized speech using the polynomial excitation was not any better than that of our earlier synthesis. Attempts will be made in the future to investigate other possible

excitation models.

#### F. Quantization Study

Currently we are planning to compare the quantization properties of several alternate sets of parameters which would uniquely characterize the linear prediction filter. Included in these are: predictor coefficients, autocorrelation coefficients of the speech signal, spectral coefficients of the linear predictor, cepstral coefficients of the linear predictor, poles of the linear predictor and the reflection coefficients. It is proposed to use the absolute error between the unquantized and the quantized predictor spectra as a criterion for evaluation. A spectral sensitivity measure is being investigated to study the sensitivity properties of these parameters under quantization.

REFERENCES

- (1) Balzer, R.M., Automatic Programming, Annual Technical Report, Information Sciences Institute, Marina del Rey, California, ISI/SR-73-1, (May 1973), 21-32.
- (2) Standish, T.A., Data Structures - An Axiomatic Approach, BBN Automatic Programming Memo 3, (August 1973)
- (3) Wegbreit, B., The Synthesis of Loop Predicates, BBN Automatic Programming Memo 2, (May 1973)
- (4) Wegbreit, B., Heuristic Methods for Mechanically Deriving Inductive Assertions, BBN Automatic Programming Memo 4, (August 1973)
- (5) Wegbreit, B., Property Extraction in Well-Founded Property Sets, BBN Automatic Programming Memo 1, (February 1973).
- (6) Rosenberg, A.E., "Effect of Glottal Shape on the Quality of Natural Vowels," The Journal of the Acoustical Society of America, Vol. 49, No. 2 (part 2), pp. 583-590 (1971)